Apprentissage d'intégrales de Sugeno à partir de données inconsistantes

Quentin Brabant, Miguel Couceiro

Apprentissage d'intégrales de Sugeno à partir de données inconsistantes

Learning a Sugeno integral from inconsistant data

Q. Brabant[1] M. Couceiro[1]
[1] LORIA (CNRS - Inria Gr,and Est - Université de Lorraine)

Équipe Orpailleur, Bâtiment B, campus scientifique
BP 239 F-54506 Vandoeuvre-lès-Nancy, France
{quentin.brabant,miguel.couceiro}@loria.fr

Résumé :
En prenant pour cadre de référence l'aide à la décision multi-critères et l'agrégation de préférences, cet article traite de l'apprentissage de l'intégrale de Sugeno à partir de données inconsistantes, et dont les valeurs appartiennent à un ensemble totalement ordonné. Il s'agit d'un problème d'optimisation difficile, puisqu'une intégrale de Sugeno est définie d'après 2^n valeurs, où n est le nombre de paramètres. Dans cet article nous considérons deux méthodes : la première est une application du recuit simulé, et la seconde est un nouvel algorithme reposant sur la séléction préalable d'un sous-ensemble de données consistantes, dont le temps d'exécution est peu sensible à la valeur de n.

Mots-clés :
Intégrale de Sugeno, optimisation, agrégation de préférence.

Abstract:
The basic setting of this article is multicriteria decision making and preference aggregation. The problem treated is that of learning a Sugeno integral from inconsistent data, where values are elements of a totally ordered set. This is a difficult optimization problem : indeed, a Sugeno integral is determined by 2^n values, with n being the number pf parameters. In this article we propose two learning methods : the first one is an application of simulated annealing, and the second is a new algorithm which relies on the selection of a consistant subset of data and for which the value of n doesn't affect the running time significantly.

Keywords:
Sugeno integral, optimization, preference aggregation.

1 Introduction

L'agrégation de préférences est une problématique appartenant au champs de l'aide à la décision multicritère (MCDA). Son objectif est de modéliser les évaluations d'un ensemble d'alternatives d'après les évaluations de chaque alternative sur plusieurs critères. Le processus par lequel des évaluations sur plusieurs critères sont combinées pour obtenir une évaluation globale est appelé *modèle d'agrégation*. Dans un contexte où les évaluations sont effectuées sur une échelle ordinale et non-numérique, l'intégrale de Sugeno est une fonction d'agrégation intéressante, puisqu'elle permet de traiter des valeurs ordinales sans les traduire en valeurs numériques.

L'intégrale de Sugeno appartient à la famille des intégrales dites floues, à laquelle appartient également l'intégrale de Choquet. Cette dernière peut être considérée comme l'analogue numérique de l'intégrale de Sugeno et est utilisée pour l'agrégation de valeurs numériques. Dans cet article nous nous intéressons au problème de l'apprentissage d'une intégrale de Sugeno à partir de données empiriques, lorsque l'échelle considérée est un ensemble fini non-numérique et totalement ordonné. Pour une description approfondie de l'intégrale de Sugeno comme fonction d'agrégation de préférences, voir par exemple [5], et pour plus d'information concernant l'utilisation des intégrales floues en MCDA, voir [7].

Soit un ensemble d'alternatives X. On modélise les préférences entre les alternatives comme une relation binaire $\preccurlyeq$ reflexive et transitive sur X, telle que $x \preccurlyeq y$ signifie "y est au moins aussi désirable que x". On considère également un ensemble L totalement ordonné,

non nécessairement numérique, que l'on appelle *échelle d'évaluation*, et dont on désigne l'élément minimal et l'élément maximal par 0 et 1 respectivement. Chaque alternative est associée à une évaluation sur cette échelle par le biais d'une *fonction d'utilité globale*

$$U : X \to L.$$

On exprime la relation de préférence entre les alternatives d'après une fonction d'utilité globale par l'équivalence

$$x \preccurlyeq y \Leftrightarrow U(x) \leq U(y).$$

Chaque alternative est evaluée sur L relativement à chaque critère d'un ensemble

$$[n] = \{1, \ldots, n\}.$$

On décrit les évaluations sur $[n]$ d'une alternative $x^j \in X$ par un tuple $\mathbf{y}^j = (y_1^j, \ldots, y_n^j)$. Les fonctions d'utilité globales que nous considérons sont celles pouvant s'exprimer comme l'agrégation des évaluations d'une alternative sur $[n]$. Plus précisément, puisque l'échelle d'évaluation est ordinale et non-numérique, la fonction d'agrégation que nous considérons est l'intégrale de Sugeno :

$$U(x^j) = \mathcal{S}_\mu(y_1^j, \ldots, y_n^j).$$

Nous rappelons la définition de l'intégrale de Sugeno (discrète) en Section 2. Dans cet article nous nous intéressons à l'apprentissage de cette intégrale à partir de données empiriques. Ce problème et ses difficultés sont discutés dans la Section 3. En Section 4 nous décrivons deux méthodes pour résoudre ce problème, à savoir une application de recuit simulé et un nouvel algorithme. Nous donnons quelques résultats d'évaluation de ces méthodes, que nous comparons en terme de temps d'exécution et de précision en Section 5. La conclusion traite des possibilités ouvertes par ces méthodes pour l'apprentissage de fonctions plus générales comme les pseudo-polynômes latticiels.

2 L'intégrale de Sugeno

L'intégrale de Sugeno a été introduite dans [11]. Nous considérons ici l'intégrale de Sugeno dans sa version discrète, appliquée à l'aggrégation de préférence. Elle se définit par rapport à une *capacité* sur l'ensemble $[n]$ qui est une fonction $\mu : 2^{[n]} \to L$ telle que

— $\mu(\varnothing) = 0$ et $\mu(X) = 1$,
— pour $I \subseteq J \subseteq [n] : \mu(I) \leq \mu(J)$.

Pour tout ensemble de critères $I \subseteq [n]$, la valeur de $\mu(I)$ peut être interprétée comme le degré d'importance associé à I.

Remarque 1. *Traditionnellement, les capacités sont définies sur un intervalle réel tel que* $[0, 1]$. *Cependant cette définition s'étend aisément à tout ensemble ordonné borné.*

Soit une capacité $\mu : 2^{[n]} \to L$. L'intégrale de Sugeno définie par rapport à μ, notée par $\mathcal{S}_\mu$, s'exprime sous la forme

$$\mathcal{S}_\mu(y_1, \ldots, y_n) = \max_{I \subseteq [n]} (\min(\mu(I), \min_{i \in I}(y_i)).$$

Par cette formule on voit notamment que μ détermine $\mathcal{S}_\mu$ entièrement et de manière unique. Le problème de l'apprentissage d'une intégrale de Sugeno $\mathcal{S}_\mu$ peut donc être ramené à celui de l'apprentissage de la capacité μ correspondante.

3 Problème d'apprentissage

On considère des données formalisées par l'ensemble

$$\mathscr{D} = \{(\mathbf{y}^1, u^1), \ldots, (\mathbf{y}^m, u^m)\} \subseteq L^n \times L,$$

dans lequel $\mathbf{y}^j = (y_1^j, \ldots, y_n^j)$ représente les évaluations par critère d'une alternative $x^j \in X$, et u^j sa valeur d'utilité globale.

On peut distinguer deux situations : dans la première, il existe $\mathcal{S}_\mu$ telle que

$$\forall j \in \{1, \ldots, m\} : \mathcal{S}_\mu(y_1^j, \ldots, y_n^j) = u^j. \quad (1)$$

Dans ce cas, l'apprentissage d'une intégrale de Sugeno repose sur la résolution du problème

d'interpolation. Ce problème est traité dans plusieurs publications, notamment [4, 10]. Les solutions à ce problème constituent l'ensemble des intégrales pour lesquelles (1) est vérifiée. Les données sont dites *inconsistantes* lorsque cet ensemble est vide.

Deux difficultées peuvent donc se poser lorsque l'on souhaite apprendre une intégrale de Sugeno à partir de nouvelles données :

— Comment déterminer efficacement si les données sont inconsistantes ?

— Lorsqu'elles le sont, comment réaliser l'apprentissage d'une intégrale de Sugeno ?

Dans [9], une méthode est présentée afin de déterminer la consistance des données (plus de détails à ce sujet seront donnés dans la Section 4). Il y est également proposé, lorsque les données sont inconsistantes, de réaliser un partitionnement de manière à ce que chaque partition soit consistante. Cependant cette méthode n'est applicable qu'à des fins descriptives, et non prédictives, puisqu'un modèle différent est associé à chaque partition.

Dans cet article, nous avons pour objectif l'apprentissage d'une intégrale de Sugeno unique aux prédictions aussi fiables que possibles. Nous mesurerons le taux d'erreur d'une intégrale de Sugeno en utilisant l'erreur d'ordonnancement paire à paire (décrite sous le nom de *pairwise error* dans [6]). Cette mesure correspond à la proportion de paires d'alternatives mal ordonnancées par le modèle d'agrégation, une paire d'alternatives $\{x^j, x^k\}$ étant dite mal ordonnancée si la relation d'ordre entre $\mathcal{S}_\mu(\mathbf{y}^j)$ et $\mathcal{S}_\mu(\mathbf{y}^k)$ diffère de celle entre u^j et u^k. Ce choix se justifie par la nature ordinale et non-numérique de L (on ne connait pas la distance entre deux valeurs de L), et par le fait que les modèles basés sur une telle échelle décrivent avant tout des relations de préférence entre les alternatives.

La minimisation du taux d'erreur d'une intégrale de Sugeno (ou, de manière équivalente, de la capacité correspondante) sur des données inconsistantes est un problème d'optimisation *a priori* complexe. Le nombre de valeurs déterminant une capacité $\mu : 2^{[n]} \to L$ croît exponentiellement avec la taille de $[n]$, c'est à dire le nombre de critères considérés (puisque l'ensemble $[n]$ possède 2^n parties). De plus, certaines méthodes comme la descente de gradient ne peuvent être appliquées, du fait de la nature non-numérique des données.

Peu de méthodes visant à résoudre ce problème d'optimisation en un temps raisonnable existent à l'heure actuelle. Nous en proposerons deux dans la partie suivante ; la première est une application du recuit simulé, tandis que la seconde est un nouvel algorithme pensé spécifiquement pour le traitement de ce problème.

4 Deux méthodes

4.1 Recuit simulé

Le recuit simulé (en anglais *simulated annealing*) est une méta-heuristique reposant sur le parcours stochastique d'un espace de solutions. Il a pour principe la modification aléatoire et successive de la solution courante, avec la possibilité de refuser les modifications qui mêneraient à de nouvelles solutions relativement désavantageuses. Chaque solution est en effet évaluée selon une fonction de coût, dont le recuit simulé a pour objectif de minimiser la valeur. La tolérance aux modifications désavantageuses dépend de l'avancement du processus, et décroît à mesure des itérations.

Pour un ensemble d'apprentissage

$$\mathscr{D}_{app} \subseteq L^n \times L,$$

nous considérons comme espace des solutions l'ensemble des intégrales de Sugeno $\mathcal{S}_\mu : L^n \to L$ (ou de manière équivalente l'ensemble des capacités $\mu : 2^{[n]} \to L$). La fonction de coût est donnée par l'erreur d'ordonnancement paire à paire de la solution considérée sur $\mathscr{D}_{app}$. À chaque itération, l'état courant est modifié de la manière suivante.

1. Tirage aléatoire d'un ensemble $I \subseteq [n]$ (chaque ensemble ayant une probabilité de 2^{-n} d'être tiré).
2. Affectation aléatoire d'une valeur $a \in L$ à $\mu(I)$ (avec une probabilité égale pour chaque valeur).
3. Répercussion du changement à la valeur de μ sur d'autres ensembles, afin de conserver la propriété de monotonicité de la capacité. On s'assure que pour tout $J \subseteq I$ on a $\mu(J) \leq \mu(I)$ et pour tout $J \supseteq I$ on a $\mu(I) \leq \mu(J)$.

La nouvelle solution ainsi produite peut être rejetée ou conservée comme nouvel état courant. La probabilité d'acceptation de la nouvelle solution est donnée par le facteur de Boltzmann :

$$P = e^{\frac{E_{\mu'} - E_{\mu}}{T}},$$

où $E_{\mu'}$ et E_{μ} désignent l'erreur d'ordonnancement paire à paire obtenue par la nouvelle solution μ' et l'ancienne solution μ, tandis que T est une variable dont la valeur décroît avec le temps, traditionnellement appelée *température*. Notez que si l'erreur de la nouvelle solution est au moins aussi performante que l'ancienne, alors $P \geq 1$. La température est initialisée à $T = 1$ et décroît linéairement à mesure des itérations jusqu'à $T = 0$. La vitesse de décroissance de la température est réglée de manière à obtenir un nombre d'itérations proportionnel au nombre de critères. Bien que le nombre de paramètres du problème soit 2^n, nous constatons en pratique qu'une augmentation linéaire du nombre d'itérations par rapport au nombre de critères suffit à donner des résultats satisfaisants en terme de précision.

Remarque 2. *La plus grande part du temps d'exécution du recuit simulé est due au calcul de l'erreur de chaque solution générée. La complexité de l'erreur d'ordonnancement paire à paire est quadratique par rapport à m, c'est pourquoi il peut être préférable de lui substituer une fonction de coût de complexité plus faible, à plus forte raison si la taille des données d'apprentissage est importante. Nous proposons d'utiliser une variante ordinale de l'erreur absolue moyenne :*

$$\frac{1}{m} \sum_{j=1}^{m} distance(\mathcal{S}_{\mu}(\boldsymbol{y}^j), u^j),$$

où la distance entre $a, b \in L$, avec $a \leq b$, est le nombre de valeurs $z \in L$ telles que $a \leq z < b$. Ceci revient à considérer arbitrairement que la distance entre deux valeurs "voisines" est de 1. Cette fonction de coût peut être calculée en temps linéaire par rapport à m, c'est pourquoi nous l'utilisons lors des tests présentés dans la Section 5.

4.2 Apprentissage par élimination

Nous proposons maintenant une nouvelle méthode d'apprentissage, dont le but est de remédier à certains inconvénients de l'application du recuit simulé, à savoir

— la nécessité de régler les paramètres de l'algorithme manuellement, notamment la vitesse de décroissance de la température,
— le temps de calcul nécessaire afin d'aboutir à une approximation satisfaisante, qui dépend à la fois du nombre de critères considérés et du nombre d'exemples traités (voir Section 5).

Nous proposons dans cette partie une méthode dont l'implémentation est aisée et ne requiert aucun choix de la part de l'utilisateur, et dont le temps d'exécution est avantageux, pour une précision presque équivalente à celle du recuit simulé (voir Section 5). La méthode que nous proposons se divise en deux parties. La première consiste à séléctionner un sous-ensemble de $\mathscr{D}$ consistant, dont la taille est idéalement maximale. La seconde est la résolution du problème d'interpolation et le choix d'une intégrale parmi l'ensemble des solutions possibles. La première partie est effectuée par l'Algorithme 1. Celui-ci est fondé sur certains résultats de [9] que nous reformulons ici de manière concise.

On dit qu'un élément $(\mathbf{y}^j, u^j) \in \mathscr{D}$ est compatible avec une intégrale de Sugeno $\mathcal{S}_\mu$ si $\mathcal{S}_\mu(\mathbf{y}^j) = u^j$. On dit que $\mathscr{D}$ est *consistant* s'il existe une intégrale de Sugeno $\mathcal{S}_\mu$ compatible avec tous les éléments de $\mathscr{D}$.

Soit $(\mathbf{y}^j, u^j) \in \mathscr{D}$. On définit deux ensembles :

$$G_j = \{i \mid y_i^j > u^j\},$$
$$G'_j = \{i \mid y_i^j \geq u^j\}.$$

Notons que $G_j \subseteq G'_j \subseteq [n]$. Une intégrale de Sugeno $\mathcal{S}_\mu$ est compatible avec $(\mathbf{y}^j, u^j)$ si et seulement si elle respecte la contrainte suivante :

$$\mu(G_j) \leq u^j \leq \mu(G'_j). \tag{2}$$

Considérons maintenant $(\mathbf{y}^j, u^j), (\mathbf{y}^k, u^k) \in \mathscr{D}$. On peut décider si ces deux éléments sont compatibles de la manière suivante.

- Si $u^j = u^k$, alors les deux éléments sont compatibles.
- Si $u^j < u^k$, alors les deux éléments sont compatibles si et seulement si $G'_k \nsubseteq G_j$.

Enfin, $\mathscr{D}$ est consistant si et seulement si tous ses éléments sont compatibles deux à deux. Disposer d'un ensemble $\mathscr{D}$ consistant est intéressant puisque dans ce cas les contraintes données par (2) décrivent l'ensemble des intégrales de Sugeno compatibles avec tous les éléments de $\mathscr{D}$.

Pour des raisons de lisibilité, nous désignerons les éléments de $\mathscr{D}$ par $d^1, \ldots, d^m$, avec $d^j = (\mathbf{y}^j, u^j)$ pour tout $j \in \{1, \ldots, m\}$.

L'Algorithme 1 s'effectue en trois étapes. La première étape (lignes 1-3) consiste à calculer G_j et G'_j pour chaque $j \in \{1, \ldots, m\}$. Durant la seconde étape (lignes 9-25), on énumère les incompatibilités présentes entre les éléments de ces données. Pour chaque élément $d^j \in \mathscr{D}$ on détermine :

- L'ensemble des éléments avec lesquels d^j est incompatible : on note cet ensemble C_j.
- Son *score d'incompatibilité* e_j, qui correspond à la cardinalité de C_j.

Les éléments sont regroupés selon leurs scores d'incompatibilité, en assignant chaque

Algorithm 1

Entrées: $\mathscr{D} = \{d^1, \ldots, d^m\}$

1: **pour tout** $j \in \{1, \ldots, m\}$ **faire**
2: calculer G_j et G'_j
3: **fin pour**
4: $I_0 \leftarrow \mathscr{D}$
5: $I_1, \ldots, I_m \leftarrow \varnothing$
6: $max \leftarrow 0$
7: $C_0, \ldots, C_n \leftarrow \varnothing$
8: $e_1, \ldots, e_m \leftarrow 0$
9: **pour tout** $j \in \{1, \ldots, m-1\}$ **faire**
10: **pour tout** $k \in \{j+1, \ldots, m\}$ **faire**
11: **si** d^j et d^k incompatibles **alors**
12: $I_{e_j} \leftarrow I_{e_j} \backslash \{d^j\}$
13: $I_{e_k} \leftarrow I_{e_k} \backslash \{d^k\}$
14: $e_j \leftarrow e_j + 1$
15: $e_k \leftarrow e_k + 1$
16: $I_{e_j} \leftarrow I_{e_j} \cup \{d^j\}$
17: $I_{e_k} \leftarrow I_{e_k} \cup \{d^k\}$
18: $C_j \leftarrow C_j \cup \{d^k\}$
19: $C_k \leftarrow C_k \cup \{d^j\}$
20: **si** $e_j > max$ ou $e_k > max$ **alors**
21: $max \leftarrow max + 1$
22: **fin si**
23: **fin si**
24: **fin pour**
25: **fin pour**
26: **tant que** $max > 0$ **faire**
27: **pour tout** $d^j \in I_{max}$ **faire**
28: **pour tout** $d^k \in C_j$ **faire**
29: **si** $e_k < e_j$ **alors**
30: $I_{e_k} \leftarrow I_{e_k} \backslash \{d^k\}$
31: $e_k \leftarrow e_k - 1$
32: $I_{e_k} \leftarrow I_{e_k} \cup \{d^k\}$
33: **fin si**
34: **fin pour**
35: $max \leftarrow max - 1$
36: **fin pour**
37: **fin tant que**
38: **retourne** I_0

élement d^j à l'ensemble I_{e_j}. La variable max contient le score d'incompatibilité maximal dans $e_1, \ldots, e_m$.

Durant la troisième étape (lignes 26-37) de l'algorithme, les éléments de $\mathscr{D}$ sont supprimés, en donnant priorité à ceux dont le nombre d'incompatibilités est le plus élevé, jusqu'à ce que tous les éléments restants soient compatibles deux à deux. Les données résultant de ces traitements sont nécessairement consistantes.

Exemple 1. *Soit une échelle d'évaluation* $\{\alpha, \beta, \gamma, \delta\}$ *telle que* $\alpha > \beta > \gamma > \delta$ *considère un ensemble* $\mathscr{D} = \{d^1, d^2, d^3, d^4\}$, *avec* $d^j = (\boldsymbol{y}^j, u^j)$ *pour* $j \in \{1, \ldots, 4\}$

$$\begin{aligned} \boldsymbol{y}^1 &= (\alpha, \alpha, \gamma), & u^1 &= \beta, \\ \boldsymbol{y}^2 &= (\alpha, \gamma, \gamma), & u^2 &= \alpha, \\ \boldsymbol{y}^3 &= (\delta, \alpha, \alpha), & u^3 &= \beta, \\ \boldsymbol{y}^4 &= (\beta, \beta, \delta), & u^4 &= \gamma. \end{aligned}$$

La première étape de l'algorithme (lignes 2-3) calcule les ensemble suivants :

$$\begin{aligned} G_1 &= \{1, 2\}, & G'_1 &= \{1, 2\}, \\ G_2 &= \varnothing, & G'_2 &= \{1\}, \\ G_3 &= \{2, 3\}, & G'_3 &= \{2, 3\}, \\ G_4 &= \{1, 2\}, & G'_4 &= \{1, 2\}. \end{aligned}$$

Ces ensembles sont ensuite utilisés durant la seconde étape de l'algorithme (lignes 9-25), afin de déterminer la compatibilité ou l'incompatibilité de chaque paire d'éléments.

$$\begin{array}{cllll} \rightarrow & \{d^1, d^2\} : & u^1 < u^2 & \textit{et} & G'_2 \subseteq G_1\,, \\ & \{d^1, d^3\} : & u^1 = u^2 & & \,, \\ & \{d^1, d^4\} : & u^4 < u^1 & \textit{mais} & G'_1 \subseteq G_4\,, \\ & \{d^2, d^3\} : & u^3 < u^2 & \textit{mais} & G'_2 \subseteq G_3\,, \\ \rightarrow & \{d^2, d^4\} : & u^4 < u^2 & \textit{et} & G'_2 \subseteq G_4\,, \\ & \{d^3, d^4\} : & u^4 < u^3 & \textit{mais} & G'_3 \subseteq G_4\,, \end{array}$$

On constate deux paires incompatibles : $\{d^1, d^2\}$ *et* $\{d^2, d^4\}$. *Puisque* d^3 *est compatible avec tous les autres éléments, on a*

$$e_3 = 0, \qquad C_3 = \varnothing.$$

Puisque d^1 *et* d^4 *sont chacun incompatibles avec* d^2 *:*

$$\begin{aligned} e_1 &= 1, & C_1 &= \{d^2\}, \\ e_4 &= 1, & C_4 &= \{d^2\}, \\ e_2 &= 2, & C_2 &= \{d^1, d^4\}. \end{aligned}$$

Les indices des éléments sont répartis dans $I_1, \ldots, I_4$ *selon leurs scores d'incompatibilité :*

$$\begin{aligned} I_0 &= \{d^1\}, \\ I_1 &= \{d^1, d^4\}, \\ I_2 &= \{d^2\}. \end{aligned}$$

Enfin, la troisième étape de l'algorithme (lignes 26-37) supprime les éléments incompatibles. Durant la première itération on a $max = 2$, $I_{max} = \{d^2\}$ *et* $C_2 = \{d^1, d^4\}$. *Le premier élément à supprimer est donc* d^2. *Le nombre d'incompatibilités dénombrées pour* d^1 *et* d^4 *sont donc décrémentés, et on obtient :*

$$e_1 = 0, \qquad e_4 = 0.$$

Par conséquent

$$I_0 = \{d^1, d^3, d^4\}, \qquad I_1 = \varnothing.$$

Après une itération, on a déterminé un sous-ensemble consistant de $\mathscr{D}$ *; l'algorithme retourne* $\{d^1, d^3, d^4\}$.

On note $\mathscr{D}_{cons}$ l'ensemble consistant obtenu en sortie de l'Algorithme 1. Pour chaque élément de D_{cons}, on connait une contrainte de la forme (2) qui caractérise l'ensemble des intégrales de Sugeno compatibles avec cet élément. L'ensemble de toutes ces contraintes caractérise un ensemble d'intégrales de Sugeno, qui est la solution du problème d'interpolation sur $\mathscr{D}_{cons}$. Chacune des intégrales de cet ensemble peut constituer une solution acceptable à notre problème d'optimisation. En pratique, nous remarquons un taux de prédiction légèrement moindre lorsque la borne inférieur de l'ensemble est utilisée comme modèle d'agrégation, c'est pourquoi nous choisissons de renvoyer cette intégrale comme résultat du processus d'apprentissage.

5 Tests de performance

Nous avons appliqué ces deux méthodes à des données empiriques [1]. Chaque élément de ces données contient l'évaluation d'un hôtel par un utilisateur du site Trip Advisor. Les évaluations sont effectuées sur une échelle discrète

$$L = \{1, 2, 3, 4, 5\},$$

et chaque hôtel est évaluation sur 7 critères, ainsi que sur sa qualité globale. Parmi les données disponibles, nous n'avons conservé que celles ne présentant pas de valeurs manquantes.

Nous avons appliqué les deux méthodes pour des tailles d'ensemble d'apprentissage variables. Pour chaque ensemble d'apprentissage, nous avons effectué une réduction du nombre d'attributs, et avons testé les deux méthodes pour des nombres variables d'attributs restants. La méthode utilisée pour la réduction du nombre d'attribut, présentée dans [1], se base sur une estimation de l'importance de chaque attribut dans la prédiction des préférences, et supprime en priorité les attributs de moindre importance.

Le Tableau 1 donne les temps moyen d'exécution (en secondes) de la méthode de recuit simulé (haut) et de l'apprentissage par élimination (bas) en fonction de la taille des données d'apprentissage et du nombre d'attributs. Ces résultats semblent indiquer que l'apprentissage par élimination s'éxécute dans un temps plus court que le recuit simulé ; cette différence est d'autant plus marquée que le nombre d'attributs est élevé et que la taille de l'ensemble d'apprentissage est basse. Comme on pouvait s'y attendre, le temps d'exécution de l'apprentissage par élimination est plus sensible au nombre d'exemples présents dans les données qu'à la valeur de n, cette dernière n'ayant pas une influence notable.

Le Tableau 2 donnc l'erreur d'ordonnancement

Tableau 1 – Temps d'exécution (s)

		Taille de $\mathscr{D}_{app}$			
		25	100	200	500
n	1	0.033	0.108	0.198	0.481
		0.001	0.017	0.059	0.346
	3	0.207	0.682	1.284	3.037
		0.002	0.016	0.055	0.332
	5	0.776	1.971	3.501	7.885
		0.002	0.017	0.057	0.346
	7	3.300	5.667	8.365	16.720
		0.004	0.020	0.061	0.353

Apprentissage par recuit simulé
Apprentissage par élimination

paire à paire moyenne obtenue par les deux méthodes en fonction du nombre d'attributs et du nombre d'exemples.

On constate que la performance des deux méthodes est influencée de manière similaire par le nombre d'attributs et la taille des données d'apprentissage. Cependant les résultats de l'apprentissage par élimination montre une précision légèrement plus faible, notamment pour des tailles d'ensembles d'apprentissage de petite taille.

6 Discussion

La méthode d'apprentissage par élimination présentée dans la Section 4.2 ouvre de nouvelles possibilités concernant l'apprentissage de pseudo-polynômes latticiels [3].

Soient $X_1, \ldots, X_n$ des ensembles quelconques. Un pseudo-polynôme latticiel $p : X_1 \times \cdots \times X_n \to L$ est une fonction pouvant être exprimée par

$$p(x_1, \ldots, x_n) = \mathcal{S}_\mu(\varphi_1(x_1), \ldots, \varphi_n(x_n)),$$

où $\mathcal{S}_\mu$ est une intégrale de Sugeno et $\varphi_1, \ldots, \varphi_n$ sont des fonctions telles que $\varphi_i : X_i \to L$. Les pseudo-polynômes latticiels peuvent être envisagés comme des modèles de préférence sur des objets décrits par la valeur de leurs attributs. Soit $\mathbf{X} = X_1, \ldots, X_n$ un ensemble d'alterna-

1. Tripadvisor Dataset : `http://sifaka.cs.uiuc.edu/~wang296/Data/index.html`.

Tableau 2 – Taux d'erreur des modèles générés

		Taille de $\mathscr{D}_{app}$			
		25	100	200	500
n	1	0.3220	0.3066	0.2938	0.2912
		0.3220	0.3066	0.2938	0.2912
	3	0.3033	0.2735	0.2547	0.2485
		0.3122	0.2769	0.2554	0.2476
	5	0.2960	0.2695	0.2588	0.2486
		0.3050	0.2697	0.2601	0.2487
	7	0.2825	0.2664	0.2600	0.2481
		0.3051	0.2755	0.2619	0.2512

Apprentissage par recuit simulé
Apprentissage par élimination

tives : chaque X_i représente alors l'ensemble des valeurs possibles du $i^{ème}$ attribut. On appelle les fonctions $\varphi_1, \ldots, \varphi_n$ des *fonctions d'utilité locales*, car elles peuvent être vues comme des fonctions d'évaluation portant chacune sur un attribut considéré indépendamment des autres.

Certains travaux présentent une axiomatisation des préférences pouvant être représentées par des pseudo-polynômes latticiels [2, 8]. Du fait de leur expressivité, les pseudo-polynômes latticiels forment une classe de fonctions intéressante du point de vue de la modélisation de préférences ; cependant leur apprentissage est un problème difficile, pusiqu'il nécessite de déterminer simultanément le modèle d'agrégation $\mathcal{S}_\mu$ et les fonctions d'utilité locales $\varphi_1, \ldots \varphi_n$.

De futures recherches devraient permettre de réaliser cet apprentissage en un temps raisonnable, en tirant parti de la méthode d'apprentissage par élimination. Ces travaux permettraient du même coup de vérifier indirectement la robustesse de l'Algorithme 1, qui ne peut être établie après un test sur un jeu de données unique. En effet, le cadre d'apprentissage d'intégrales de Sugeno est relativement restrictif, puisqu'il suppose une description de chaque alternative sur L^n : en pratique, les données satisfaisant à cette contrainte sont rares. Les pseudo-polynômes latticiels en revanche, permettent de modéliser les préférences d'objets dont la forme de la description est moins restrictive. Par conséquent les méthodes visant à leur apprentissage devraient pouvoir être testées sur un plus grand nombre de données.

Références

[1] Q. Brabant, M. Couceiro, F. Labernia, A. Napoli. Une approche de réduction de dimensionnalité pour l'agrégation de préférences qualitatives. *Actes de la 16ème Conférence francophone sur l'Extraction et la Gestion des Connaissances, 2016*, pp. 345-350.

[2] D. Bouyssou, T. Marchant, M. Pirlot. A conjoint measurement approach to the discrete Sugeno integral. *The Mathematics of Preference, Choice and Order.* Springer Berlin Heidelberg, 85-109, 2009.

[3] M. Couceiro, T. Waldhauser. Pseudo-polynomial functions over finite distributive lattices, *Fuzzy Sets and Systems*, 161(5) : 694–707, 2010.

[4] M. Couceiro, D. Dubois, H. Prade, A. Rico, T. Waldhauser. General interpolation by polynomial functions of distributive lattices. *Information Processing and Management of Uncertainty in Knowledge-Based Systems, Communications in Computer and Information Science*, vol. 299, Springer-Verlag, 347-355, 2012.

[5] D. Dubois, J-L. Marichal, H. Prade, M. Roubens, R. Sabbadin. The use of the discrete Sugeno integral in decision making : a survey. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(5) : 539-561, 2001.

[6] J. Fürnkranz, E. Hüllermeier. *Preference learning*. Springer, 2011.

[7] M. Grabisch, C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *4OR : A Quarterly Journal of Operations Research*, 6(1) : 1-44, 2008.

[8] S. Greco, B. Matarazzo, R. Słowiński. Axiomatic characterization of a general utility function and its particular cases in terms of conjoint measurement and rough-set decision rules. *European Journal of Operational Research,* 158(2) : 271-292, 2004.

[9] H. Prade, A. Rico, M. Serrurier, E. Raufaste. Elicitating Sugeno integrals : Methodology and a case study. *Actes de la 10ème Conférence Européenne sur les Approches Quantitative et Symbolique du Raisonnement et de l'Incertitude (ESCQARU), 2009*, pp. 712-723.

[10] A. Rico, M. Grabisch, C. Labreuche, A. Chateauneuf. Preference modelling on totally ordered sets by the Sugeno integral. *Discrete Applied Mathematics*, 147(1) : 113-124, 2005.

[11] M. Sugeno, *Theory of fuzzy integrals and its applications.* Thèse doctorale, Tokyo Institute of Technology, 1974.